WL-TR-92-1055

1

A STEP-BY-STEP GUIDE FOR PRODUCING ACEC REPORTS USING THE DEC VAX ADA COMPILER

Major Stephen A. Davidson, USAF Reserves
Software Concepts Group
System Avionics Division

WRIGHT LABORATORY

30 March 1992

Final Report for Period 15 July 1991 to 26 July 1991

AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-6543

This technical report has been reviewed and is approved for publication.

Steven A. Davidson

STEVEN A. DAVIDSON, Major
U.S. Air Force Reserves

Robert L. Harris

ROBERT L. HARRIS, Group Chief
Software Concepts Group
Avionics Logistics Branch

FOR THE COMMANDER

Donna M Morris

DONNA M. MORRIS, Chief
Avionics Logistics Branch
Systems Avionics Division

Accesion For	
NTIS CRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution unlimited	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S) WL-TM-91-128		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Wright Laboratory	6b. OFFICE SYMBOL (If applicable) WL/AAAF	7a. NAME OF MONITORING ORGANIZATION WL/AAAF	
6c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6543		7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6543	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION WL/ASD (AFSC) Avionics Directorate	8b. OFFICE SYMBOL (If applicable) WL/AAAF-3	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER In-House	
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6543		10. SOURCE OF FUNDING NUMBERS	

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.
63756D	2853	01	28530103

11. TITLE (Include Security Classification)
A STEP-BY-STEP GUIDE FOR PRODUCING ACEC REPORTS USING THE DEC VAX ADA COMPILER

12. PERSONAL AUTHOR(S)
Maj Steven A. Davidson, USAF Reserves

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Summary	FROM 91-7-15 TO 91-7-26	91-11-08	29

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Compiler Evaluation, Ada Language, DEC VAX Useage, Software Tools

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report provides explicit, step-by-step instructions on how to execute the Ada Compiler Evaluation Capability software evaluation tool on a Digital Equipment Corporation VAX computer. It provides the reader a "worked example" of execution to help use this tool more quickly; and to improve the user's understanding of the information provided in the tool's User Guide.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Robert L. Harris	22b. TELEPHONE (Include Area Code) (513)255-3947	22c. OFFICE SYMBOL WL/AAAF-3

FOREWORD

This report documents work completed July 1991 under the Aeronautical Systems Division's Air Force Reserves project 90-432-LAB. The work was largely accomplished during the two-week period of 15-26 July 1991. It provides a guide for using the Ada Compiler Evaluation Capability (ACEC) software tool residing at WL/AAAF on its VAX/4000 cluster.

It was the intent of this work to provide an easier access to the ACEC, for researchers within the Avionics Logistics Branch (WL/AAAF), by documenting its use based on "hands-on" experience. This report is particularly valuable for helping to solve the considerable "mechanics" associated with using this enormously important tool.

Robert L. Harris

Robert L. Harris, Chief

Software Concepts Group

WL/AAAF-3

TABLE OF CONTENTS

Section 1

INTRODUCTION

1. Background.

The Ada Compiler Evaluation Capability (ACEC) is a software tool developed by Boeing Military Airplanes (BMA) for the Avionics Logistics Branch of the Avionics Directorate of Wright Laboratory (WL/AAAF) at Wright-Patterson Air Force Base, Dayton, Ohio. The ACEC measures the performance of an Ada compiler hosted on a specific computer. The ACEC can compile and execute over 1100 Ada test routines which exercise different features of the Ada language compiler. Compilation speed, compiled code memory size, and execution time are measured. The ACEC user can use this information to compare performance between compiler/computer combinations, isolate strengths and weaknesses of a compiler/computer pair, and determine changes between releases of a compiler (1).

2. The ACEC User's Guide.

BMA wrote a User's Guide (2) to help the user execute the ACEC. The present version of the ACEC is difficult to execute. It consists of numerous programs, and batch control files which must be edited, compiled, and executed in a a specific sequence to obtain the desired results. The User's Guide describes how to execute ACEC on a "generic computer". It instructs the ACEC user in general terms. However, the User's Guide does not provide explicit, step-by-step instructions for executing the ACEC on a specific computer system.

3. Purpose.

The purpose of this paper is to provide explicit, step-by-step instructions on how to execute the ACEC on a Digital Equipment Corporation (DEC) VAX main frame computer. This provides the reader with a "worked example" of ACEC execution which will help the reader use ACEC more quickly and improve the reader's understanding of the information provided in the User's Guide.

4. Assumptions.

This paper makes the following assumptions:

a. Users Guide. The reader is expected to have a copy of the User's Guide for ACEC, as described in Reference 2. This paper makes frequent references to this document.

b. DEC Command Language (DCL). The reader is expected to be somewhat familiar with DCL commands used on VAX computers. All DCL commands necessary to obtain results are provided in this paper. However, users who are not comfortable with DCL are advised to obtain and refer to a DCL user's manual to better understand the DCL commands used in this paper.

c. Ada Compilation System (acs). The .COM files on the ACEC distribution tape are written such that the Ada compiler, linker, and library are invoked by means of the VAX DCL utility "acs". This utility software must be available on the VAX machine being used to compile the Ada test routines and the analysis routines (such as MEDIAN, or SSA).

d. Text File Editor. The reader is expected to be able to use an on-line text editor to modify or create VAX command files (.COM) written in ASCII text. The VAX text file screen editors such as EDT or the Language Sensitive Editor (LSE) are recommended. However, any text file editor (or word processor) capable of reading and writing ASCII text files will do.

e. Ada Compiler Language. The reader is not required to be familiar with the Ada Compiler Language to exercise ACEC using the steps listed in this paper.

5. **Format Conventions.**

This paper will use the following format conventions to promote understandability and minimize confusion:

a. Interactive On-Line Commands. Commands which are to be entered at an on-line VAX terminal are indented and set apart from the supporting text of each step by blank lines. Each command appears on a separate line. The end of the line implies the carriage return (RETURN) key is to be pressed. However, a few exceptions to this convention occur when the command line is too long to fit, as a single line, on the page of this document. For those situations, the remaining portion of the command will be printed immediately below and indented one space from the initial portion of the command printed on the line above. An example of a long, single command which is printed in this report on two lines is:

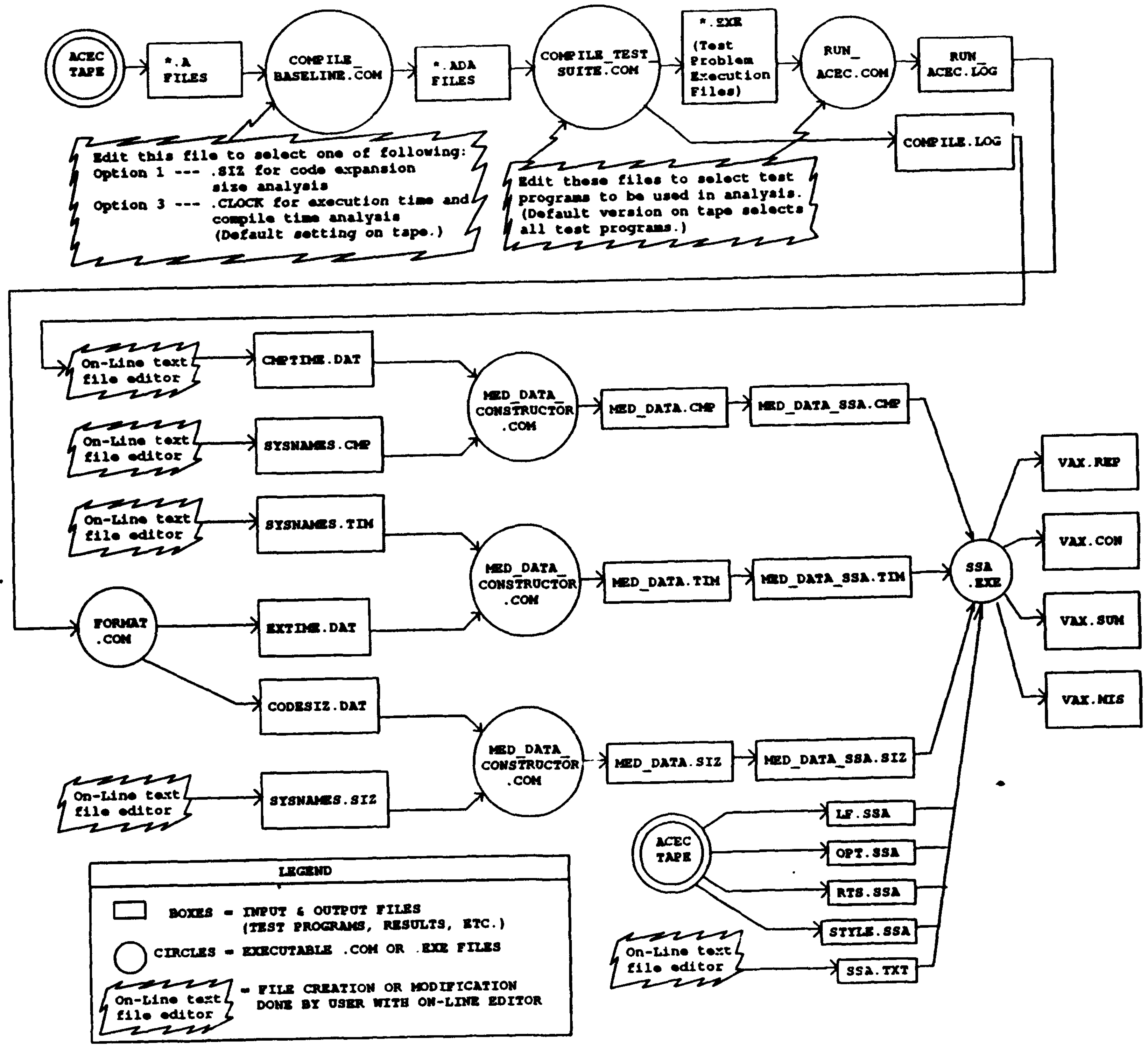
```
submit/notify/log_file=compile.log/after=17:00
 compile_acec.com
```

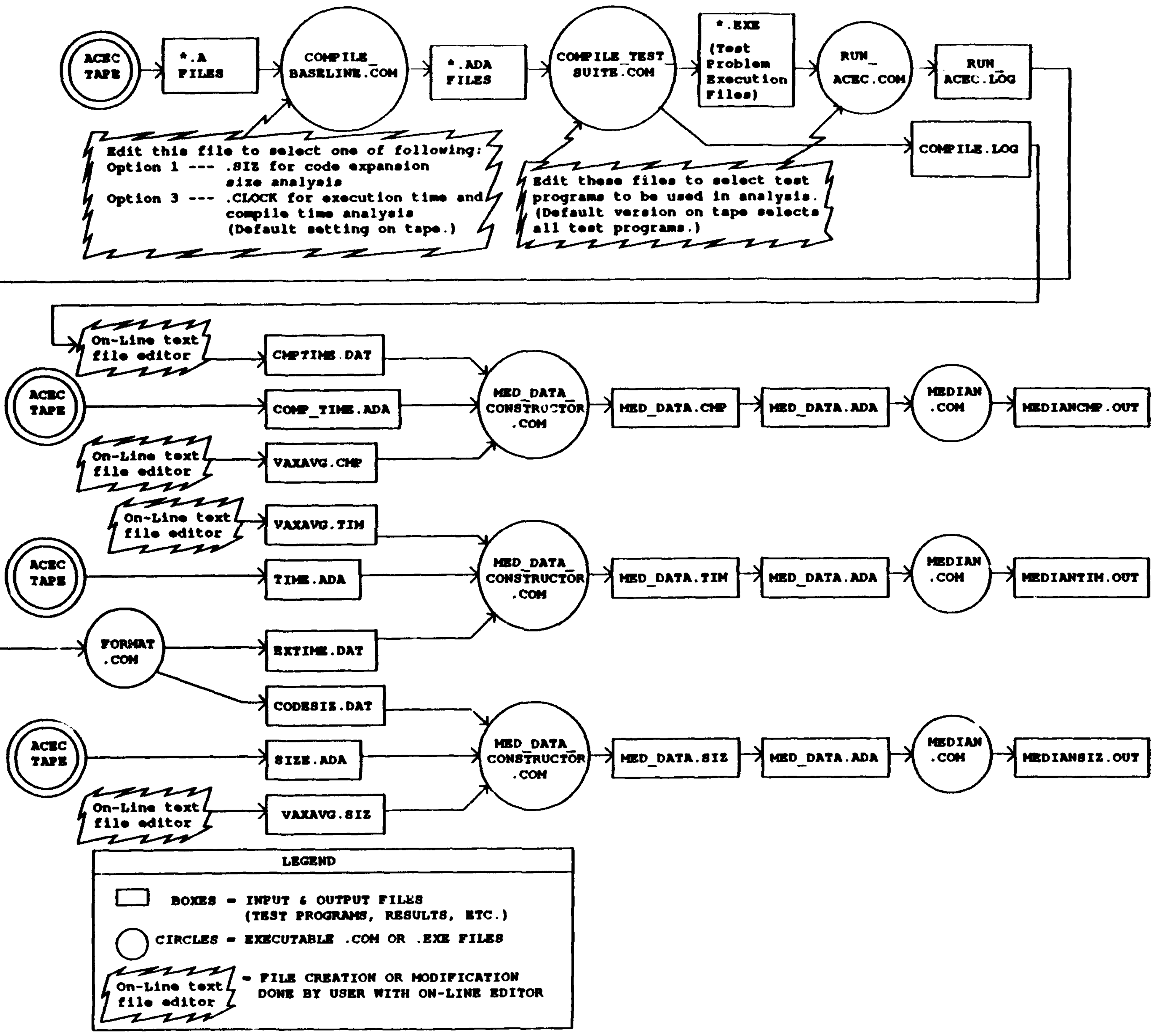
In this example, instead of pressing the RETURN key at the end of the first line (after "17:00), press the space bar for one space and continue typing the next indented line (starting with "compile_" as a single command line on the terminal. Then press the RETURN key at the end of the indented line (after ".COM") to complete transmission of the command.

b. File Names. File names discussed in the text will be typed in capitals (example: MEDIAN.COM).

6. Flow Charts.

Figures 1 and 2 are flow charts which diagram the relationships among the Ada test program (.A and .ADA) files, the executive (.EXE) files, the batch command (.COM) files, and the desired data and output files. The flow charts also suggest the sequence of steps the user must follow to obtain results. It is very helpful to examine these flow charts while reading the following sections. Sections 2 and 3 are applicable to the upper half of both flow charts, which are the same. Section 4 is applicable to the lower half of Figure 1 and Section 5 is applicable to the lower half of Figure 2.


ACEC TAPE
*.A FILES
COMPILE_BASELINE.COM
*.ADA FILES
COMPILE_TEST_SUITE.COM
*.EXE (Test Problem Execution Files)
RUN_ACEC.COM
RUN_ACEC.LOG
COMPILE.LOG
Edit this file to select one of following:
Option 1 --- .SIZ for code expansion size analysis
Option 3 --- .CLOCK for execution time and compile time analysis (Default setting on tape.)
Edit these files to select test programs to be used in analysis. (Default version on tape selects all test programs.)
On-Line text file editor
CMPTIME.DAT
SYSNAMES.CMP
SYSNAMES.TIM
FORMAT .COM
EXTIME.DAT
CODESIZ.DAT
SYSNAMES.SIZ
MED_DATA_CONSTRUCTOR .COM
MED_DATA.CMP
MED_DATA_SSA.CMP
MED_DATA.TIM
MED_DATA_SSA.TIM
MED_DATA.SIZ
MED_DATA_SSA.SIZ
SSA .EXE
VAX.REP
VAX.CON
VAX.SUM
VAX.MIS
LF.SSA
OPT.SSA
RTS.SSA
STYLE.SSA
SSA.TXT
LEGEND
BOXES = INPUT & OUTPUT FILES (TEST PROGRAMS, RESULTS, ETC.)
CIRCLES = EXECUTABLE .COM OR .EXE FILES
= FILE CREATION OR MODIFICATION DONE BY USER WITH ON-LINE EDITOR


ACEC TAPE
*.A FILES
COMPILE_ BASELINE.COM
*.ADA FILES
COMPILE_TEST_ SUITE.COM
*.EXE (Test Problem Execution Files)
RUN_ ACEC.COM
RUN_ ACEC.LOG
COMPILE.LOG
Edit this file to select one of following:
Option 1 --- .SIZ for code expansion size analysis
Option 3 --- .CLOCK for execution time and compile time analysis (Default setting on tape.)
Edit these files to select test programs to be used in analysis. (Default version on tape selects all test programs.)
On-Line text file editor
CMPTIME.DAT
ACEC TAPE
COMP_TIME.ADA
On-Line text file editor
VAXAVG.CMP
MED_DATA_ CONSTRUCTOR .COM
MED_DATA.CMP
MED_DATA.ADA
MEDIAN .COM
MEDIANCMP.OUT
On-Line text file editor
VAXAVG.TIM
ACEC TAPE
TIME.ADA
FORMAT .COM
EXTIME.DAT
MED_DATA_ CONSTRUCTOR .COM
MED_DATA.TIM
MED_DATA.ADA
MEDIAN .COM
MEDIANTIM.OUT
CODESIZ.DAT
ACEC TAPE
SIZE.ADA
On-Line text file editor
VAXAVG.SIZ
MED_DATA_ CONSTRUCTOR .COM
MED_DATA.SIZ
MED_DATA.ADA
MEDIAN .COM
MEDIANSIZ.OUT
LEGEND
BOXES - INPUT & OUTPUT FILES (TEST PROGRAMS, RESULTS, ETC.)
CIRCLES - EXECUTABLE .COM OR .EXE FILES
On-Line text file editor - FILE CREATION OR MODIFICATION DONE BY USER WITH ON-LINE EDITOR

Section 2

INITIAL SET-UP STEPS

1. Step 1 - Set Up Directories and Read ACEC Distribution Tape.

First, a structure of sub-directories should be set up in the user's area to facilitate better organization of files. It also helps the user's get a better understanding of ACEC program execution. The steps in this paper for executing ACEC are based on the following directory and sub-directory structure. The top directory is [davidson]. Of course, the reader's VAX account name would be substituted for "davidson" in all the commands presented in this paper. The sub-directories are set up by entering the VAX commands:

```
       create/dir [davidson.acec]
       create/dir [davidson.acec.work]
```

Sub-directory [davidson.acec] contains the ACEC program files which must be read in from the distribution tape or copied from someone else's account who has already read the distribution tape (see your VAX System Manager). Files in this directory are not modified or executed. All modifications and executions are done in the work area directory [davidson.acec.work]. This area is considered a "scratch pad" area and is initially empty.

2. Step 2 - Create the Working Ada Library.

The Ada library used by the VAX Ada compiler is located in [davidson.acec.work.adalib] and MUST be created using the acs command by entering:

```
        acs create library [davidson.acec.work.adalib]
```

3. Step 3 - Copy Files Into Work Area.

Copy the following files into [davidson.acec.work]. Go to the [.acec] directory by entering:

```
        set def [davidson.acec]
```

Enter the copy commands:

```
       copy *.a [.work]*
       copy *.ada [.work]*
       copy *.com [.work]*
       copy *.clock [.work]*
       copy *.port [.work]*
```

```
        copy ran32.ada [.work]*
        copy math_dependent.dec [.work]*
        copy sysnames.txt [.work]*
```

4. Step 4 - Invoke the Ada Library.

Identify the current Ada library. First, go down to the [.work] directory (from the [davidson.acec] directory). Enter:

```
        set def [.work]
```

Then identify the current Ada library by entering:

```
    acs set lib [.adalib]
```

5. Step 5 - Identify Default Work Directory in Primary .COM Files

Using the Language Sensitive Editor (LSE) editor (or any editor you like on the VAX), modify the following .COM files so that the correct local directories are specified and the Ada Library is identified.

```
        COMPILE_BASELINE.COM
        COMPILE_ACEC.COM
        COMPILE_TEST_SUITE.COM
        RUN_ACEC.COM
        MEDIAN.COM
        MED_DATA_CONSTRUCTOR.COM
        COMPILE_RUN_ACEC.COM
```

For example, based on the directory structure discussed in Steps 1 and 2, the following 2 lines should be added (or modified) to the above .COM files:

```
        set def [davidson.acec.work]
        acs set library [davidson.acec.work.adalib]
```

These lines are always within the first two or three lines of the .COM file. The .COM file, COMPILE_RUN_ACEC.COM, is a .COM file I wrote and is not available from the ACEC tape. It is explained later and is not needed until Step 49.

6. Step 6 - Convert Ada Test Routines Into Compilable Units.

Run COMPILE_BASELINE.COM to compile the package GLOBAL and package MATH. (See page 18 of the User's Guide.) Enter:

```
    @compile_baseline
```

Running COMPILE_BASELINE.COM also modifies every Ada test routine on the distribution tape with a file extension ".A", into a test routine with a file extension ".ADA", which will later be compiled. Each Ada test routine is modified to add statements to the routine which will either measure execution time and compile time, if OPTION 3 is used (the default) or measure code expansion size if OPTION 1 is selected. Steps 46 through 48 describe how to switch to and use OPTION 1 in COMPILE_BASELINE.COM . (Initially, use the default setting, OPTION 3; do not modify COMPILE_BASELINE.COM .

7. Step 7 - Math and Time Measurement Tests of the Ada Compiler.

a. Run SETUP_TEST_PROGRAMS.COM to compile and link MATHTEST, DBL_MATHTEST, TESTCAL1, and TESTCAL2. (See page 18 of the User's Guide.) SETUP_TEST_PROGRAMS.COM will also execute MATHTEST and DBL_MATHTEST. Enter:

```
@setup_test_programs
```

The results of executing MATHTEST and DBL_MATHTEST will rapidly flash by on the terminal screen. To get a hard-copy of the results of this program (which checks the math functions of the Ada compiler under test), the following command can be entered at the terminal **after** SETUP_TEST_PROGRAMS.COM has been executed:

```
@run_test_programs/output=vaxtest.out
```

Results will be written to the file VAXTEST.OUT in the [.work] directory. You can then view this file using an editor or route the file to a printer. The printed results is 35 pages long.

b. The accuracy of the system clock routines used by the Ada Compiler under test is checked **after** SETUP_TEST_PROGRAMS.COM is executed, by running TESTCAL1.EXE and TESTCAL2.EXE interactively from an on-line VAX terminal.

Note: These programs each take 15 minutes of time to execute and I was unable to interrupt or terminate their execution during this 15-minute period. (In other words, don't run these programs if you're in a hurry to get other things done.)

Execute TESTCAL1.EXE by entering:

```
run testcal1
```

Follow the directions on the screen. Similarly, TESTCAL2.EXE is executed by entering:

```
run testcal2
```

8. Step 8 - Compile Utility Programs INCLUDE and FORMAT.

Run COMPILE_TOOLS.COM to compile and link INCLUDE.ADA and FORMAT.ADA . This MUST be done (once) before the test suites are compiled using COMPILE_TEST_SUITE.COM the first time. Enter:

```
@compile_tools
```

Section 3

COMPILING AND EXECUTING THE TEST SUITE

1. Step 9 - Compile the Ada Test Routines.

a. Compile all the test programs using COMPILE_TEST_SUITE.COM . This will take about **6.5 hours** to run on the AAAF VAX; run it as a batch job!!! Execution can be delayed until after 5 pm by adding the "/after" clause to the submit command. Enter (on a single line):

```
       submit/notify/log_file=compile.log/after=17:00
        compile_test_suite.com
```

Results will be written to the file COMPILE.LOG in the top directory (in this case, [davidson]), after compilation is completed. COMPILE.LOG is necessary for the analysis programs MEDIAN and SSA to provide information about compilation time of the test programs. So, if you are interested in compilation time analysis, don't delete COMPILE.LOG! Of course, COMPILE_TEST_SUITE.COM will also generate the object files for each test program and place them in the Ada library [davidson.acec.work.adalib] and the executable *.EXE files will be written to the [.work] directory. That's why it's important to add the statements:

```
set def [davidson.acec.work]
acs set lib [davidson.acec.work.adalib]
```

to the first lines of COMPILE_TEST_SUIT.COM so that the Ada compiler will know where to find the Ada files (created by INCLUDE.ADA from the *.A files), and where to write the Ada object files (in the Ada library) and the executable *.EXE files in the [.work] directory.

b. After becoming more familiar with ACEC, you may want to perform additional analysis using only selected Ada test routines. It should be possible to edit COMPILE_TEST_SUITE.COM to "comment-out" compilation of all unwanted routines. (I have not yet tried to do this.)

2. Step 10 - Alternative: Convert and Compile Ada Test Routines In One Step.

If Steps 6 though 9 have been performed, Step 10 is not necessary. However, Step 10 is discussed as an alternative method to performing Steps 6 through 9, (after the user has a better understanding of how ACEC works.) The .COM file COMPILE_ACEC.COM will perform most of the tasks in Steps 6 through 9 above, automatically. Because this .COM file runs COMPILE_TEST_SUITE.COM, COMPILE_ACEC will also take about **6.5 hours** to run on the AAAF VAX and should therefore be run only as a batch job. To do this, enter (on a single line):

```
submit/notify/log_file=compile.log/after=17:00
 compile_acec.com
```

Note that after Steps 6 and 8 have been done once, for a given compiler, they do not need to be repeated. Also, there will be no hard-copy results of MATHTEST and DBL_MATHTEST in Step 7 by using COMPILE_ACEC.COM .

3. Step 11 - Move the COMPILE.LOG Output File To the Work Directory.

After the batch job has completed execution, the COMPILE.LOG file must be moved from the top directory to the [.work] directory. A VAX command which can do this in a single step is:

```
rename [davidson]compile.log [davidson.acec.work]*.*
```

4. Step 12 - Delete all ".A" Files in the Work Directory.

To conserve file space and tidy things up, you can delete all the *.A files in the [.work] directory, but DO NOT delete the *.A files in the [.acec] directory.

delete [davidson.acec.work]*.A; *

5. Step 13 - Check COMPILE.LOG Output File For Error Messages.

The COMPILE.LOG file will contain over 31,000 lines of output. DON'T PRINT IT! However, you may want to check there are no (serious) compilation errors by searching for the word "error" in COMPILE.LOG using an editor or by entering the following VAX command:

```
search/output=cmp_errors.dat/window=7 compile.log error
```

This will find all occasions of the word "error" in the file and will write the line with "error" in it plus 3 lines before and 3 lines after (for a total 'window' of 7 lines) to the output file CMP_ERRORS.DAT . Then CMP_ERRORS.DAT can be viewed with an editor or routed to a printer for a hard-copy.

6. Step 14 - Execute the Suite of Ada Test Routines.

a. RUN_ACEC.COM will execute the test programs which have been previously compiled by COMPILE_TEST_SUITE.COM in Step 9 or

COMPILE_ACEC.COM in Step 10. This will take **8.5 hours** to execute so it MUST also be run as a batch job!!! Execution can be delayed until after 5 pm by adding the "/after" clause to the submit command. Enter:

```
submit/notify/after=17:00 run_acec.com
```

Results will be written to the file RUN_ACEC.LOG in the top directory (in this case, [davidson]), after compilation is completed. It's important to add the statements:

```
set def [davidson.acec.work]
acs set lib [davidson.acec.work.adalib]
```

to assure RUN_ACEC.LOG will find all executable files for the test programs created by Step 9 or Step 10 in the [.work] directory.

b. If COMPILE_TEST_SUITE.COM was edited to "comment-out" selected Ada test routines as discussed in Step 9, these same Ada test routines must also be "commented-out" of RUN_ACEC.COM to prevent errors caused by attempting to execute routines which were not compiled.

7. Step 15 - Check RUN_ACEC.LOG Output File For Error Messages.

The RUN_ACEC.LOG file will contain nearly 14,000 lines of output. DON'T PRINT IT! However, *you may* want to check there are no (serious) execution errors by searching for the word "error" in RUN_ACEC.LOG using an editor or by entering the following VAX command:

```
search/output=exec_errors.dat/window=7 run_acec.log error
```

This will find all occasions of the word "error" in the file and will write the line with "error" in it plus 3 lines before and 3 lines after (for a total 'window' of 7 lines) to the output file EXEC_ERRORS.DAT . Then EXEC_ERRORS.DAT can be viewed with an editor or routed to a printer for a hard-copy.

8. Step 16 - Move RUN_ACEC.LOG to the Work Directory.

After the batch job has executed, the RUN_ACEC.LOG file must be moved from the top directory to the [.work] directory. A VAX command which can do this in a single step is:

```
rename [davidson]run_acec.log [davidson.acec.work]*.*
```

9. Step 17 - Produce EXTIME.DAT and CODESIZ.DAT Data Files.

Run FORMAT.COM using RUN_ACEC.LOG as the input file to produce the execution time data file and the code expansion size data file. (See Page 84 of the User's Guide.)

```
@format run_acec.log extime.dat codesiz.dat
```

The execution time information will be written to the EXTIME.DAT file and the code expansion size data will be written to the CODESIZ.DAT file. Choice of these output file names is arbitrary but once chosen, these file names must be used consistently for the information they represent in the steps which follow.

Section 4

RUNNING SSA

1. Purpose and Requirements of SSA.

a. The Single System Analysis (SSA) program analyzes relationships between test problems executed on one Ada compiler system (see Page 83 of the User's Guide). To execute without errors, SSA requires three input data files which contain information about the test programs compiled in Step 9 (or Step 10) and executed in Step 14. (See Page 96 of the User's Guide.)

b. Use MED_DATA_CONSTRUCTOR.COM to generate two of the files needed for SSA. These files to be generated are MED_DATA.TIM and MED_DATA.SIZ .

2. Step 18 - Identify the Default Work Directory In MED_DATA_CONSTRUCTOR.COM

First, edit MED_DATA_CONSTRUCTOR.COM to verify (or add) the statements at the very beginning of the .COM file:

```
set def [davidson.acec.work]
acs set lib [davidson.acec.work.adalib]
```

This will assure the Ada library is set (in case you just logged on again). Also, it assures execution will occur on files in the [.work] directory.

3. Step 19 - Create SYSNAMES.TIM

To produce MED_DATA.TIM, edit the file SYSNAMES.TXT to generate SYSNAMES.TIM . (See Page 86 of the User's Guide.) The file SYSNAMES.TIM should consist of the following lines of text:

```
time    -- or "size" or "compile"
vax     -- name of compiler system under analysis
--
--  Ada compiler on AAAF VAX
--  as of July 18, 1991
extime.dat
```

As with Ada, any text written after the "--" symbol is treated as a comment. Up to 20 lines of comments may be added after the second line which specifies the "system-name". The system name "vax" on the second line is arbitrary but once chosen, this name must be used consistently to represent the "system-name" in the steps which follow. The next line after the comments MUST be the file name for execution time file (EXTIME.DAT) which was created by FORMAT.COM in Step 17.

4. Step 20 - Produce MED_DATA.TIM

Generate MED_DATA.TIM by entering the command:

```
@med_data_constructor sysnames.tim
```

The file name parameter "sysnames.tim" MUST be the name of the file created in Step 19.

5. Step 21 - Create SYSNAMES.SIZ

To produce MED_DATA.SIZ edit the file SYSNAMES.TIM to generate SYSNAMES.SIZ . (See Page 86 of the User's Guide.) The file SYSNAMES.SIZ should consist of the following lines of text:

```
size   -- or "time" or "compile"
vax    -- name of compiler system under analysis
--
--  Ada compiler on AAAF VAX
--  as of July 18, 1991
codesiz.dat
```

As in Step 19, any text written after the "--" symbol is treated as a comment. Up to 20 lines of comments may be added after the second line which specifies the "system-name". The system name "vax" on the second line must be the same as was selected in Step 19. The next line after the comments MUST be the file name for the code expansion size file (CODESIZ.DAT) which was created by FORMAT.COM in Step 17.

6. Step 22 - Produce MED_DATA.SIZ

Generate MED_DATA.SIZ by entering the command:

```
@med_data_constructor sysnames.siz
```

The file name parameter "sysnames.siz" MUST be the name of the file created in Step 19.

7. Step 23 - Create CMPTIME.DAT

a. The third file needed for SSA is MED_DATA.CMP, the compilation time data file. It can't be generated by using MED_DATA_CONSTRUCTOR.COM with an input file from FORMAT.COM because FORMAT.COM generates only the two files previously discussed in Step 17. (See Page 85 of the User's Guide.) As a result, MED_DATA.CMP must be created using MED_DATA_CONSTRUCTOR.COM which, in turn, uses a text data input file, CMPTIME.DAT, that is constructed manually using information in file COMPILE.LOG . The format for file CMPTIME.DAT, will be similar to the format of EXTIME.DAT or CODESIZ.DAT and is explained on Pages 96 through 98 of the User's Guide. The following steps represent one way to build this file which appears to yield reasonable results.

b. Step 24. To build the CMPTIME.DAT file, first strip out all the text lines in COMPILE.LOG, created in Step 9, which contain information on compilation time. These lines contain the symbol "=>" . The following VAX command can be used to create a file consisting of the desired information:

```
search/output=cmptime.dat compile.log
```

Using a text editor such as LSE, view the file CMPTIME.DAT which has just been created. Notice that every line begins with the text string:

```
"compile and link time in hundredths of a second for  "
```

which is followed by the test program name, the "=>" symbol, and an integer number. Next, strip out the text string on every line so that only the test program name, "=>" symbol, and integer number remain. This can be easily done with most editors by substituting a null string for the text string shown above. Within the LSE editor, enter the command (at the LSE> prompt, all on one line):

```
substitute "compile and link time in hundredths of a second
 for  " ""
```

The result should be that every line in file CMPTIME.DAT contains a test program name, beginning in Column 1. For example:

```
ACKER2 => 3674
ARTI => 17676
CFA => 5071
    .
    .
    .
```

c. Step 25. Add a comment line as a new first line of the file to describe the file contents:

```
-- compile and link time in seconds
```

d. Step 26. On what is now the second line, insert the 'start-of-information symbol', "(", symbol prior to the name of the first test program listed. For example, if the first test program name listed is ACKER2, the second line will now appear as:

```
(ACKER2 => 3674
```

e. Step 27. Insert a decimal point in front of the least 2 significant digits for <u>every</u> number in the file. For example, the lines listed in Step 25 become:

```
(ACKER2 => 36.74
ARTI => 176.76
CFA => 50.71
    .
    .
    .
```

This changes hundredths of seconds to seconds. It also changes the numbers from integers to floating point numbers which MUST be done to prevent an Ada exception (error) caused by a data type mismatch when MED_DATA-CONSTRUCTOR.COM is executed.

f. Step 28. Add the 'end-of-information' symbol, ")", after the last number in the file, at the end of the file. For example, if the last test program name listed is TRIE, the last line in the file will be:

```
TRIE => 53.70)
```

Note: It is not necessary to add commas to the ends of the lines in file CMPTIME.DAT .

8. Step 29 - Create SYSNAMES.CMP

To produce MED_DATA.CMP, edit the file SYSNAMES.TIM to generate SYSNAMES.CMP . (See Page 86 of the User's Guide.) The file SYSNAMES.CMP should consist of the following lines of text:

```
compile   -- or "time" or "size"
vax    -- name of compiler system under analysis
--
--  Ada compiler on AAAF VAX
--  as of July 18, 1991
cmptime.dat
```

As in Step 19, any text written after the "--" symbol is treated as a comment. Up to 20 lines of comments may be added after the second line which specifies the "system-name". The system name "vax" on the second line must be the same as was selected in Step 19. The next line after the comments MUST be the file name for the compilation time file, (CMPTIME.DAT), which was created in

Steps 23 through 28.

9. Step 30 - Produce MED_DATA.CMP

Generate MED_DATA.CMP by entering the command:

```
@med_data_constructor sysnames.cmp
```

The file name parameter "sysnames.cmp" MUST be the name of the file created in Step 29.

10. Step 31 - Delete Unnecessary files In the Work Directory.

Steps 20, 22, and 30 will also generate three VAX versions of the files:

```
MED_DATA_CONSTRUCTOR.EXE
MED_DATA_CONSTRUCTOR.LIS
MDC_ERRORS.TXT
```

If all goes well and MED_DATA.TIM and MED_DATA.SIZ were produced and look reasonable, these other files listed above can be deleted. (MED_DATA.SIZ will contain all zeros initially but this will not cause an execution error. Steps 46 through 52 show how to get non-zero values for this file.)

```
delete [davidson.acec.work]med_data_constructor.exe;*
delete [davidson.acec.work]med_data_constructor.lis;*
delete [davidson.acec.work]mdc_errors.txt;*
```

11. Step 32 - Rename MED_DATA_CONSTRUCTOR Output Files To Prevent Confusion.

Rename the three files now created by MED_DATA_CONSTRUCTOR.COM to prevent confusion later when MED_DATA_CONSTRUCTOR.COM is used to produce similar files for MEDIAN.COM . Rename the following files:

```
MED_DATA.TIM
MED_DATA.SIZ
MED_DATA.CMP
```

to the new file names:

```
MED_DATA_SSA.TIM
MED_DATA_SSA.SIZ
MED_DATA_SSA.CMP
```

This can be easily done (at this point in the initial sequence of steps) by using the following single VAX command:

```
rename med_data.* med_data_ssa.*
```

12. Step 33 - Create SSA.TXT

Using an editor, create the file SSA.TXT . Insert the following two lines of text (starting in Column 1):

```
s=vax
i=med_data_ssa.tim, med_data_ssa.siz, med_data_ssa.cmp
```

The value after the "s=" MUST be the same system name as appears in the second line of the MED_DATA_SSA.* files. The second line of file SSA.TXT specifies the input files for SSA which have been created in Steps 20, 22, and 30. The order of the files is important.

13. Step 34 - Compile SSA

Compile and link SSA.ADA . This is more easily done using a .COM file, such as COMPILE_SSA.COM which I wrote. File COMPILE_SSA.COM consists of the following lines of text (starting in Column 1):

```
$ set verify
$ set def [davidson.acec.work]
$ acs set lib [davidson.acec.work.adalib]
$ ada/nowarn/nocheck/optimize=time ssa
$ acs link ssa
```

Execute COMPILE_SSA.COM by entering:

```
@COMPILE_SSA
```

14. Step 35 - Execute SSA

a. Prepare to execute SSA. Copy over the four input template files required by SSA.EXE from the [.acec] directory into the [.work] directory. (See Page 99 of the User's Guide.) These four files are:

```
LF.SSA
OPT.SSA
RTS.SSA
STYLE.SSA
```

This can be easily done by entering the single VAX command:

```
copy [davidson.acec]*.ssa [davidson.acec.work]*.*
```

b. Verify the following files are now in the [.work] directory:

```
SSA.EXE
SSA.TXT
MED_DATA_SSA.TIM
MED_DATA_SSA.SIZ
MED_DATA_SSA.CMP
LF.SSA
OPT.SSA
RTS.SSA
STYLE.SSA
```

This can be done by entering the following two VAX commands from within the [.work] directory:

```
dir *ssa*.*
dir *.ssa
```

Take appropriate action if any of these files are not in the [.work] directory.

c. Execute SSA.EXE by entering:

```
run ssa
```

If SSA executes properly, it will produce at least three output files with a file name equal to the name of the Ada Compiler System specified in the input file SSA.TXT (see Step 33) and an extension name of .REP, .CON, or .SUM . A fourth file with the extension .MIS is produced if the flag for the missing data report is set to "true". This is done by adding a third line, "m=true", to the SSA.TXT file (see Page 92 through 96 of the User's Guide). In our case, the system name is "vax" so the following reports are produced:

```
VAX.REP
VAX.CON
VAX.SUM
```

VAX.REP, the body of the SSA report, is a very large file. If routed to a printer, over 200 pages of text will be printed. The user should consider viewing the file with an editor first, to assure the appropriate analysis was performed, before routing this file to the printer. VAX.CON is a table of contents for the main report in VAX.REP and VAX.SUM is a summary of the report in VAX.REP. If routed to a printer, about 7 and 16 pages of text will be printed, respectively.

Section 5

RUNNING MEDIAN

1. Purpose and Requirements of MEDIAN.

The MEDIAN analysis program compares sets of performance data from at least two different compilation systems (see Page 83 of the User's Guide). If there is only one Ada compiler system available, the data files provided on the ACEC distribution tape, which contain the average of the performance of five trial systems analyzed during the development of ACEC, can be used in place of a second actual system Ada compiler (see Page 87 of the User's Guide). The User's Guide implies this information is contained in one file:

DATA.ADA

In truth, this average-of-performance information is contained in three separate files:

TIME.ADA
SIZE.ADA
COMP_TIME.ADA

These files are similar to the files produced by FORMAT.COM, as discussed in Step 17 (EXTIME.DAT and CODESIZ.DAT for TIME.ADA and SIZE.ADA) and the compilation time data file built by hand in Step 23 through Step 28 (CMPTIME.DAT for COMP_TIME.ADA).

2. Step 36 - Copy the Average-of-Performance Files To the Work Directory.

Copy the files discussed in the preceding paragraph from the [.acec] directory to the [.work] directory. This can be done by entering the VAX commands:

```
copy [davidson.acec]time.ada [davidson.acec.work]*
copy [davidson.acec]size.ada [davidson.acec.work]*
copy [davidson.acec]comp_time.ada [davidson.acec.work]*
```

3. Step 37 - Identify Three Types of Analysis Results of MEDIAN.

Use the MED_DATA_CONSTRUCTOR to generate new MED_DATA.* files that will contain 2 Ada compiler systems, VAX and the Average of 5 Ada compilers from the data files discussed in Step 36. Depending on the type of analysis desired (execution time, code expansion size, or compilation time), MEDIAN requires at least one of the following three input files to be prepared which contain performance data on (at least) two systems. These three files are:

MED_DATA.TIM	(for execution time analysis)
MED_DATA.SIZ	(for code expansion size analysis)
MED_DATA.CMP	(for compilation time analysis)

4. MEDIAN - Execution Time Analysis.

For execution time analysis, follow Step 38 through Step 41. First, it will be necessary to prepare a new SYSNAMES.TXT-type file similar to the ones discussed in Step 19, Step 21, or Step 29.

a. Step 38 - Create VAXAVG.TIM . To produce MED_DATA.TIM, edit the file SYSNAMES.TXT to generate VAXAVG.TIM . (See Page 86 of the User's Guide.) The file VAXAVG.TIM should consist of the following lines of text:

```
time    -- or "size" or "compile"
vax     -- name of System 1 under analysis
--
--  Ada compiler on AAAF VAX
--  as of July 18, 1991
extime.dat
average  -- name of System 2
--
--      Average of 5 trial Ada compilers
time.ada
```

As with Ada, any text written after the "--" symbol is treated as a comment. Up to 20 lines of comments may be added after the second line which specifies the "system-name". The system name "vax" on the second line is arbitrary but once chosen, this name must be used consistently to represent the "system-name" in the steps which follow. The next line after the comments MUST be the file name for execution time file which was created by FORMAT.COM in Step 17 (EXTIME.DAT). The next line after the execution time file for System 1 must be the System 2 name (average) to be compared to System 1 (vax). Up to 20 more comment lines can follow. The next non-comment line MUST be the file name for the execution time file for System 2 (TIME.ADA) which was either created by FORMAT.COM (for another system compiler tested) or, as in this case, copied as a data file (TIME.ADA) from the ACEC distribution tape.

b. Step 39 - Produce MED_DATA.TIM . Generate MED_DATA.TIM by entering the command:

```
          @med_data_constructor vaxavg.tim
```

The file name parameter "vaxavg.tim" MUST be the name of the file created in Step 37.

c. Step 40 - Copy MED_DATA.TIM Into MED_DATA.ADA . To produce the execution time analysis report using MEDIAN, copy MED_DATA.TIM to a "dummy" file called MED_DATA.ADA using the VAX command:

```
          copy med_data.time med_data.ada
```

d. Step 41 - Execute MEDIAN to Obtain Execution Time Analysis. Now execute MEDIAN to produce the execution time report by entering the VAX command:

```
          @median mediantim.out
```

MEDIAN will read the execution time input data from the "dummy" file MED_DATA.ADA . The report will be written to the output file, MEDIANTIM.OUT, which may be viewed using the editor or printed. The printed report is over 50 printed pages long. (But also, a blank page appears between every printed page, causing the printer to produce over 100 pages of output! This is a program/printer compatibility problem which should be corrected.)

5. MEDIAN - Compilation Time Analysis.

For compilation time analysis, follow Step 42 through Step 45.

a. Step 42 - Create VAXAVG.CMP . To produce MED_DATA.CMP, edit the file VAXAVG.TIM, produced in Step 38, to generate VAXAVG.CMP . (See Page 86 of the User's Guide.) The file VAXAVG.CMP should consist of the following lines of text:

```
compile   -- or "time" or "size"
vax    -- name of System 1 under analysis
--
--  Ada compiler on AAAF VAX
--  as of July 18, 1991
cmptime.dat
average  -- name of System 2
--
--      Average of 5 trial Ada compilers
comp_time.ada
```

As in Step 38, any text written after the "--" symbol is treated as a comment. Up to 20 lines of comments may be added after the second line which specifies the "system-name". The system name "vax" on the second line must be the same as was selected in Step 37. The next line after the comments MUST be the file name for the compilation time file (CMPTIME.DAT) which was created in Steps 23 through 28. The next line after the compilation time file for System 1 must be the System 2 name (average) to be compared to System 1 (vax). Up to 20 more comment lines can follow. The next non-comment line MUST be the file name for the compilation time file for System 2 (COMP_TIME.ADA) which was either created by following Steps 23 through 28 (for another system compiler tested) or, as in this case, copied as a data file (COMP_TIME.ADA) from the ACEC distribution tape.

b. Step 43 - Produce MED_DATA.CMP. Generate MED_DATA.CMP by entering the command:

```
@med_data_constructor vaxavg.cmp
```

The file name parameter "vaxavg.cmp" MUST be the name of the file created in Step 42.

c. Step 44 - Copy MED_DATA.CMP Into MED_DATA.ADA . To produce the compilation time analysis report using MEDIAN, copy MED_DATA.TIM to the "dummy" file called MED_DATA.ADA using the VAX command:

```
copy med_data.cmp med_data.ada
```

d. Step 45 - Execute MEDIAN to Obtain Compilation Time Analysis. Now execute MEDIAN to produce the compilation time report by entering the VAX command:

```
@median mediancmp.out
```

MEDIAN will read the compilation time input data from the "dummy" file, MED_DATA.ADA . The report will be written to the output file, MEDIANCMP.OUT, which may be viewed using the editor or printed. The printed report is over 50 printed pages long. (But also, a blank page appears between every printed page, causing the printer to produce over 100 pages of output! This is a program/printer compatibility problem which should be corrected.)

6. MEDIAN - Code Expansion size Analysis.

For code expansion size analysis, follow Step 46 through Step 55. Ideally, the Steps 38 through 41 could be modified appropriately, as was done in Steps 42 through 45 for compilation time analysis, to obtain code expansion size analysis. Unfortunately, for the DEC Ada Compiler on the AAAF VAX, additional "work-around" steps are necessary. This is because in Versions 1.5 and earlier of the DEC Ada, the label'ADDRESS attribute (of the Ada Language) incorrectly always returns the value zero. ACEC relies on the label'ADDRESS attribute to measure code expansion sizes. As a result, using ACEC in its default mode, demonstrated in the previous 45 steps, yields all zeros for code expansion size information. (See Pages 52, 53, 62, and 63 of the User's Guide. Also, the reader can view files RUN_ACEC.LOG, CODESIZ.DAT created by FORMAT.COM in Step 17, or MED_DATA_SSA.SIZ created by Steps 21, 22, and 32, to verify this point.) As a "work-around", a utility assembler function, GETADR.MAR on the ACEC distribution tape, can be used to obtain code addresses, in place of the label'ADDRESS attribute of the Ada Language. However, as the User's Guide explains on Page 63, this "work-around" affects the execution time measurements, making them unreliable.

a. Step 46 - Identify the Need to Re-compile Ada Test Routines. To summarize, the User can not obtain both (reliable) execution time analysis and code expansion size analysis from a single execution of ACEC. After execution time data and analysis has been performed, if code expansion size data and analysis is desired, the COMPILE_BASELINE.COM file must be modified and all the ACEC test programs must be re-compiled and re-executed. Steps 47 through 53 demonstrate how to do this.

b. Step 47 - Copy Additional Utility Routines Into the Work Area. Copy the following distribution tape files with file extension .SIZ in the [.acec] directory to the [.work] directory by entering the following VAX commands:

```
copy [davidson.acec]inittime.siz [davidson.acec.work]*.*
copy [davidson.acec]startime.siz [davidson.acec.work]*.*
copy [davidson.acec]stoptime0.siz [davidson.acec.work]*.*
copy [davidson.acec]stoptime2.siz [davidson.acec.work]*.*
copy [davidson.acec]global.siz [davidson.acec.work]*.*
```

Also copy the assembler function with extension .MAR by entering:

```
copy [davidson.acec]getadr.mar [davidson.acec.work]*.*
```

c. Step 48 - Edit and Execute COMPILE_BASELINE.COM To Select "OPTION 1". Using an editor, modify the file COMPILE_BASELINE.COM to "comment-out" statements for 'OPTION 3 --- .CLOCK' by changing "$ " to "$!" in Columns 1 and 2 of these lines. Next, "un-comment" the command statements for 'OPTION 1 --- .SIZ', changing "$! " to "$ " by deleting the "!" without replacement. There are 12 command lines and one comment line which remains a comment line. The comment line begins with the date "$! 18 December 1989, ... ".

d. Step 49 - Re-compile and Execute All Ada Test Routines. Copy the Ada test routines with the ".A" extension back into the work directory:

```
copy [davidson.acec]*.a [davidson.acec.work]*
```

Re-run COMPILE_BASELINE.COM to modify the Ada test routines into compilable units as was done in Step 6. This will produce a new set of compilable Ada test routines with the ".ADA" extension. Compile and run the entire ACEC suite again by submitting a .COM file. I wrote a special .COM file to compile and execute ACEC in a single over-night session (which runs about **15 hours** on the AAAF VAX computer. The .COM file I wrote is called COMPILE_RUN_ACEC.COM and is submitted by entering:

```
    submit/notify/after=17:00 compile_run_acec.com
```

The compile time results will be written to the file COMPILE_RUN_ACEC.LOG and the execution and code expansion size results will be written to the file RUN_ACEC.LOG. Both files will be in the top directory ([davidson]) the next day.

The following is a copy of COMPILE_RUN_ACEC.COM which I wrote:

```
$ set verify
$ set def [davidson.acec.work]
$ acs set lib [davidson.acec.work.adalib]
$!
$!
------------------------------------------------------------------------
$!  Rename the directory [davidson.acec.work] here
$!  and in all other .COM files to a local directory
$!  usable for testing.
$!
$!  This .COM file will compile and run all
$!  ACEC test problems.  If the full set of test problems
$!  is used (normally done, by default), the Main-Frame
$!  VAX (in AAAF) takes about 6.5 hours to compile all test
$!  programs using COMPILE_TEST_SUITE.COM and about
$!  8.5 hours to execute all test programs using
$!  RUN_ACEC.COM .
$!
$!  Needless to say, this should be done overnight
$!  by submitting this .COM file as a batch job after 5 pm!!!
$!  This is done by entering the following command
$!  at the terminal:
$!
$!      submit/notify/after=17:00 compile_run_acec.com
$!
$!
------------------------------------------------------------------------
$!
$!  First compile the test suite and write the result to
$!  file COMPILE_RUN_ACEC.LOG in the top directory.
$!
```

```
$ @compile_test_suite
$!
$!  Next, execute the test suite.  The run-time results will
$!  be written to file RUN_ACEC.LOG in the top directory.
$!
$ submit/notify RUN_ACEC.COM
$!
```

e. Step 50 - Move RUN_ACEC.LOG to the Work Directory. Move RUN_ACEC.LOG to the [.work] directory using the VAX command:

```
rename [davidson]run_acec.log [davidson.acec.work]*.*
```

f. Step 51 - Produce EXTIME.DAT and CODESIZ.DAT Data Files. Run FORMAT.COM again, as was done in Step 17 to re-generate CODESIZ.DAT from the new RUN_ACEC.LOG file which should now have non-zero values for the code expansion size for every test program. Enter the VAX commands:

```
set def [davidson.acec.work]
@format run_acec.log extime.log codesiz.dat
```

g. Step 52 - Create VAXAVG.SIZ. To produce MED_DATA.CMP, edit the file VAXAVG.TIM, produced in Step 38, to generate VAXAVG.SIZ . (See Page 86 of the User's Guide.) The file VAXAVG.SIZ should consist of the following lines of text:

```
size    -- or "time" or "compile"
vax     -- name of System 1 under analysis
--
--  Ada compiler on AAAF VAX
--  as of July 18, 1991
codesiz.dat
average  -- name of System 2
--
--      Average of 5 trial Ada compilers
size.ada
```

As in Step 38, any text written after the "--" symbol is treated as a comment. Up to 20 lines of comments may be added after the second line which specifies the "system-name". The system name "vax" on the second line must be the same as was selected in Step 37. The next line after the comments MUST be the file name for the code expansion size file (CODESIZ.DAT) which was created in Step 51. The next line after the code expansion size file for System 1 must be the System 2 name (average) to be compared to System 1 (vax). Up to 20 more comment lines can follow. The next non-comment line MUST be the file name for the code expansion size file for System 2 (SIZE.ADA) which was either created by FORMAT.COM (for another system compiler tested) or, as in this case, copied as a data file (SIZE.ADA) from the ACEC distribution tape.

h. Step 53 - Produce MED_DATA.SIZ. Generate MED_DATA.SIZ, similar to Step 43, by entering the VAX command:

```
@med_data_constructor vaxavg.siz
```

i. Step 54 - Copy MED_DATA.SIZ Into MED_DATA.ADA. To produce the code expansion size report using MEDIAN, copy MED_DATA.SIZ to the "dummy" file called MED_DATA.ADA by entering the VAX command:

```
copy med_data.siz med_data.ada
```

j. Step 55 - Execute MEDIAN To Obtain Code Expansion Size Analysis. Now execute MEDIAN using code expansion size data in MED_DATA.ADA by entering the VAX command:

```
@median mediansize.out
```

The report will be written to the file MEDIANSIZE.OUT which can be viewed using an editor or printed. Even though this is memory size (in terms of bytes or bits), the resulting report still indicates the values are a measure of <u>time</u>. This is confusing and should be corrected.

7. Step 56 - Delete Unnecessary Files In the Work Directory.

Steps 41, 45, and 55 will also generate three sets of the files:

```
MED_DATA_CONSTRUCTOR.EXE
MED_DATA_CONSTRUCTOR.LIS
MDC_ERRORS.TXT
```

If all goes well and MED_DATA.TIM and MED_DATA.SIZ were produced and look reasonable, these other files listed above can be deleted.

```
delete [davidson.acec.work]med_data_constructor.exe;*
delete [davidson.aced.work]med_data_constructor.lis;*
delete [davidson.acec.work]mdc_errors.txt;*
```

References

1. Tom Leavitt, Kermit Terrell, Barbara Deeker-Lindsey, Ada Compiler Evaluation Capability (ACEC), an Overview (Draft), Boeing Military Airplanes, Wichita, Kansas, April 24, 1989.

2. Tom Leavitt, Ada Compiler Evaluation Capability (ACEC), Technical Operating Report (TOR), User's Guide Release 2.0, Boeing Military Airplanes, Wichita, Kansas, D500-12470-1, February 8, 1990.